Linked data schemata: fixing unsound foundations.

Kevin Feeney, Gavin Mendel Gleason, Rob Brennan

Knowledge and Data Engineering Group & ADAPT Centre, School of Computer Science & Statistics, Trinity College Dublin, Ireland

Abstract. This paper describes an analysis, and the tools and methods used to produce it, of the practical and logical implications of unifying common linked data vocabularies into a single logical model. In order to support any type of reasoning or even just simple type-checking, the vocabularies that are referenced by linked data statements need to be unified into a complete model wherever they reference or reuse terms that have been defined in other linked data vocabularies. Strong interdependencies between vocabularies are common and a large number of logical and practical problems make this unification inconsistent and messy. However, the situation is far from hopeless. We identify a minimal set of necessary fixes that can be carried out to make a large number of widely-deployed vocabularies mutually compatible, and a set of wider-ranging recommendations for linked data ontology design best practice to help alleviate the problem in future. Finally we make some suggestions for improving OWL's support for distributed authoring and ontology reuse in the wild.

Keywords: Linked Data, Reasoning, Data Quality

1. Introduction

One of the central tenets of the Linked Data movement is the reuse of terms from existing well-known vocabularies [Bizer09] when developing new schemata or datasets. The semantic web infrastructure, and the RDF, RDFS and OWL languages, support this with their inherently distributed and modular nature. In practice, through vocabulary reuse, linked data schemata adopt knowledge models that are based on multiple, independently devised ontologies that often exhibit varying definitional semantics [Hogan12]. In order to correctly construct these heterogeneous schemata, in order to reason about them or to validate the contents of linked data datasets that reference them, a basic requirement is the ability to create a unified knowledge model which combines the referenced ontologies and vocabularies.

There are, however, significant theoretical and practical problems in creating a sound and consistent logical model from the vocabularies which are typically utilised in linked data. For example linked data often freely mixes references to ontologies defined in OWL and vocabularies defined in RDFS. As OWL is a syntactic but not semantic extension of RDF, there are significant well-understood difficulties in creating any unification between RDF models and OWL models [Hor03]. Beyond the theoretical problems, all systems where components are developed independently from one another and later combined face significant coordination challenges [Heineman01]. For example the well-known *ontology hijacking problem*, most often caused by misuse of OWL's equivalence statements [Hogan10], occurs when a statement in one vocabulary modifies an entity defined in another. Given OWL's limited support for modularity [Patel02], we should expect such problems to occur in practice. However, although such problems are well known in theory, there has been little work in systematically assessing their manifestations in published Linked Data, and how to deal with them in practice. This is largely a consequence of the lack of tools which can help to identify the problems. The open world semantics applied in standard OWL reasoners are so permissive as to produce valid models for a variety of obvious specification errors, such as misspellings and typos [Patel02]. Vocabulary specifications are, in many cases, sufficiently long and complex that manual analysis is difficult, time-consuming and error prone, rendering it impractical at scale.

In this paper we investigate the following research questions: (1) to what extent can current heterogeneous linked data vocabularies be unified into consistent logical models? (2) What useful logical or syntactic errors can we detect and correct in building these models? (3) What is the distribution of logical or syntactical schemata errors present in the current web of data?

In order to address these, and other related questions, we constructed a *Dacura Quality Service*[1]. This service consumes OWL and RDF linked data schemata specifications, unifies them into common models, wherever their structures require it by term reuse, and analyses them with a specially constructed reasoner to identify potential problems in their specifications. This reasoner uses a much less permissive interpretation than that of standard OWL-DL, to find issues which are likely to stem from specification errors, even in cases where they produce valid OWL models. This tool is integrated into a general purpose ontology analysis framework in the Dacura platform [Feeney14] which identifies structural dependencies between ontologies, incidences of ontology hijacking, and a range of heuristic quality measures.

This paper's contributions are: the identification and analysis of the practical and theoretical challenges encountered when unifying the models of linked data schemata observed in the current web of data; a description of the Dacura Quality Service approach to model unification; an extensive quality evaluation of linked data vocabularies in common use on the web of data, for logical and syntactical errors; and finally a set of recommendations on best practice for constructing linked data vocabularies that could produce compatible logical models, which can be more easily unified, given the distributed authoring environment of the web.

The structure of the rest of this paper is as follows: in section 2 we discuss the challenges for linked data schema validation, in section 3 we discuss related work, in section 4, the Dacura Quality Service is described, with a focus on its schema validation capabilities. Section 5 describes a wide-scale experimental validation of linked data vocabularies conducted with the Dacura Quality Service and the methodology that was used. The results of this evaluation are presented in section 6. Section 7 present a set of recommendations for best practice in linked data vocabulary design and specification, and finally, section 8 describes our conclusions and the future direction of this work.

2. Challenges for Linked Data Schemata Validation

We define a *linked data schema* as the formal description of the structure of a linked data dataset, expressed in RDF, RDFS and/or OWL vocabularies or ontologies, that is sufficiently complete that all individuals in the dataset are described in terms of a consistent logical model of their classes, properties or datatypes. Thus, there are no unspecified terms used in the schema and it is possible to unify all the definitions into a single logical model that respects the specification semantics of the component vocabularies. This enables validation: as all terms used must be defined, the definitions must not lead to inconsistency and for some use cases the definitions form the basis for integrity constraints on data described by the schema. However, real world linked data schemata are often described by multiple, interdependent vocabularies with heterogeneous specification languages, definitional semantics, authors and publishers, which leads to many challenges when determining the definition of all terms or building a unified logical model.

Challenge 1: Heterogeneous Use of RDF, RDFS, OWL and others

OWL DL ontologies describe a formal domain model based on description logic. It is a difficult task to produce a logical model which accurately and correctly encapsulates any non-trivial domain [Hepp07]. This has probably influenced the relative popularity of RDFS, with its weaker semantics, in

[1] Source code available at https://github.com/GavinMendelGleason/dacura

linked data [Polleres13]. In the wild, RDFS and OWL are mixed very freely [Hogan12], [Polleres13]. Polleres et al. describe this as their "challenge 2" for reasoning over linked data i.e. that it is not pure OWL. In fact, some common linked data vocabularies make reference to other ontologies which are entirely incompatible with OWL: specified in raw RDF and using RDF collection types, or in DAML or some other esoteric language (see figure 2 below for an outline of the evidence we collected). Since these ontologies reference each other's terms, full validation cannot proceed without determining whether the referenced ontologies are themselves consistent and complete.

If linked data was limited to the use of RDFS or OWL DL, or perhaps even some extension of OWL DL which could encompass elements of OWL Full (such as predication over classes and properties) then consistent model checking would be possible. However the problematic historical unification of RDF and OWL as OWL Full has led to an interpretative fissure between it and OWL DL [Hor03]. OWL DL provides a clear model theoretic semantics which allows one to decidably determine whether a given OWL ontology, potentially coupled with instance data, is consistent. By contrast, OWL Full mixes in the very loose syntactic rules of RDF to arrive at a compromise between OWL and RDF. It is not decidable due to mixing of logical and metalogical symbols. Ultimately, the full unification of RDF and OWL was dropped as a requirement for OWL Full in OWL2 [Grau08].

There are two particularly problematic deficiencies encountered when attempting to interpret RDF/RDFS ontologies as OWL for the purpose of model unification.

The first is the use of primitive RDF collection types. The primitive RDF properties rdf:first, and rdf:next are encountered in linked data vocabularies "in the wild" but these are used as internal syntactic symbols by OWL. They cannot be treated as language elements at the level of properties and classes without introducing inconsistency.

The second problem relates to predication. In OWL DL, one may not refer to classes of classes, or properties whose domains are themselves classes or properties. This ensures decidability and in necessary to avoid well-known "Russell-type" paradoxes such as this one derived from [Patel02].

```
ex:noResources a owl:Restriction .
ex:noResources owl:onProperty rdf:type ;
ex:noResources owl:onClass ex:hasAResource ;
ex:noResources owl:maxQualifiedCardinality "0"^^xsd:nonNegativeInteger ;
ex:hasAResource oneOf ( ex:noResources ) .
```

This particular OWL description is satisfied only when it is not, and vice versa. The difficulty arises from the ability to quantify *naively* over the rdf:type property itself. This is very similar to Russell's use of the set of all sets. There are methods, well known to logicians [Luo90], of allowing predication over classes and predicates by introducing stratification to the quantification, but no syntax to do so is present in OWL.

In summary the heterogeneity of linked data schemata means that OWL Full is insufficient for validation and the incompatibilities between RDF and OWL DL mean that even if a single model could be constructed, OWL Full would be undecidable and/or incomplete.

Challenge 2: Linked Data Import and Referencing Semantics

In linked data schemata there are two ways that other ontologies/vocabularies are referenced, either by explicitly including them using owl:imports, or implicitly by making reference to URIs of properties and classes in their namespace. The meaning of the first is given a precise semantics under OWL DL (which is not unproblematic in its own right as we will see later): the entire imported ontology is unioned with the current one during reasoning. The second, where URIs are referred without importation, has no formal semantics but is a widely used convention (for example see

[Courtot09]). This leads to the question of how to validate over such opaque references. This is a serious problem as one could potentially be referring to an instance as a class, or a class as an instance, one could have references to a class which refers to a third ontology which is shared and does not allow sound subsumption, or any number of other such problematic mixing of meanings without any well-defined way to check for correctness.

Challenge 3: The Impact of Distributed Authoring and Publication

Developing ontologies that can be easily reused in contexts that were not anticipated by the ontology developer is analogous to the software engineering challenge of developing libraries for reuse in situations where they must coexist with a wide variety of other libraries. A basic principle of software engineering is that libraries which use other libraries should not change their behaviour for other libraries. Similarly, ontologies which alter other ontologies are dangerous. Gruber expressed one aspect of this as being "able to define new terms for special uses based on the existing vocabulary, in a way that does not require the revision of the existing definitions" [Gruber93]. This sensitivity to ontological redefinition is particularly relevant as OWL's support for modularity is extremely primitive – the *import* statement unifies models into a common model that has a global scope. This means that any ontology, A, that modifies another ontology, B, effectively changes ontology B for all other ontologies that use it and in a distributed environment this alteration is both silent and invisible to other ontologies that reference B, but do not use ontology A. Of course if A, B and a third ontology that uses B are subsequently unified into a single model, then logical inconsistencies can become apparent. This problem has been referred to as ontology hijacking [Hogan10]. Due to the complexity of OWL, inconsistencies arising from such problems may only be detectable by automated reasoners. However, despite the prevalence of OWL terms in linked data vocabularies, it is evident from our findings, that many vocabulary authors do not perform reasoner-based checks prior to publication.

Linked data's focus on the reuse of independently developed and maintained ontologies introduces other significant practical problems. Ontologies that reuse other ontologies are vulnerable to these referenced ontologies becoming unavailable over time, or changing in ways that render them incompatible [Polleres13]. This highlights the weaknesses in OWL and especially RDFS's ontology or vocabulary lifecycle support and the variety of practices observed makes automated approaches untenable for the open web of data where many core vocabularies predate even OWL2's limited versioning meta-data. Given the wide diversity of contexts in which they have been developed – and the cost and difficulty in maintaining them – there is a significant risk of ontologies degenerating over time due to changes in the availability or structure of their dependent ontologies.

Challenge 4: Permissivity of OWL

OWL is an extremely permissive language – OWL reasoners will create a valid model wherever possible, inferring many elements automatically [OWL2Semantics]. Thus a number of OWL descriptions which are formally correct contain human errors not intended by the ontology designer, and yet will produce valid models. For example, the following assertions:

ex:name rdfs:domain ex:Bear, ex:Pig;
ex:peter a ex:Man;
ex:peter ex:name "Peter".

These will create Peter as an instance of a "ManBearPig" due to OWL allowing inference of class axioms that would produce a valid model. This is counter-intuitive to software engineers who assume closed world semantics and a class structure that must be declared in advance. Many of those publishing linked data schemata are not expert knowledge engineers and hence such mistakes

proliferate. Even among experts who understand the logical structure of the language well, the complexity of many ontologies and the inferences that are applied to them is such that these specification errors occur commonly in practice yet they are not detected by standard reasoners.

3. Related Work

There is a wide variety of existing research that is relevant to our research questions, we categorise it here under three main headings: (1) frameworks and approaches for assessing linked data quality, (2) theoretical studies on the unification of RDF and OWL and (3) reasoning and consuming linked data research. Each of these is discussed in turn in the subsections below.

Frameworks and approaches for assessing linked data quality

The underlying framework for current linked data quality assessment has been defined by Zalveri et al. [Zaveri15]. In terms of their conceptual framework, our current work addresses mainly intrinsic dimensions of the schema – syntactic validity, semantic accuracy (in terms of misuse of properties) and consistency. However our quality service also covers some of their contextual dimensions, for example, by checking for human-readable labelling of properties and classes, it addresses understandibility.

Our current work builds upon the previous version of our Dacura data curation platform [Feeney14] by extending the simple rule-based data validation implemented in Apache Jena/Java, described in the previous publication, with a custom reasoner and ACID triplestore for validation and data integrity enforcement. This new component, the Dacura Quality Service, is built in SWI-Prolog on ClioPatria [Wielemaker15] and is described in the next section. An earlier version of the Dacura Quality Service which covered a much smaller set of OWL features was described in an earlier paper [MG15]. That work has been extended here to also include a discussion of the new Dacura Schema Management service, our experimental validation of linked data schemata in the wild, and new recommendations for best practice when constructing new linked data vocabularies.

The RDFUnit methodology for test-driven quality assessment by Kontokostas et al. [Konto14] is a SPARQL-based approach to validating linked data schemata and datasets. It is described by Zaveri et al. [Zaveri15] as being able to detect intrinsic data quality dimensions for syntactic and semantic accuracy, but in common with all SPARQL-based approaches the lack of reasoning ability means that it is difficult to automatically detect consistency problems. For a specific dataset, it is possible to manually generate specific SPARQL-based tests that could detect these errors, but the manual effort required is significant and the approach is brittle in the presence of schemata change over time. Similar approaches have been taken with SPARQL and SPIN (SPARQL Inferencing Notation) [Furber10] and the Pellet Integrity Constraint Validator (ICV) [Sirin09].

Since 2014 the W3C's Data Shapes Working Group[2] has been working on SHACL (Shapes Constraint Language) as a language for describing structural constraints against which RDF instance data can be validated. Instance data validation, which is the primary use case of the SHACL working group, is not the focus of this paper. However, it is a closely related problem: OWL does not distinguish at a language level between instance and schema. OWL terms such as owl:oneOf are associated with instance data, but are frequently used to provide type enumeration, for example, within schema specifications. It is possible that suitable SHACL constraints could be used to validate RDF graphs describing schemata. However, this would – at a minimum – require restating the existing logical models expressed in OWL, with models expressed in another specification language which happens to have incompatible semantics (SHACL). In the long-run, to be successful, the SHACL approach

[2] https://www.w3.org/2014/data-shapes/wiki/Main_Page

requires the long-term maintenance of logical equivalence between models expressed in different and incompatible formalisms. Given the difficulty in maintaining consistency between linked data schemata and instance data, it is hard to imagine that an approach that depends on the much more challenging task of maintaining model equivalence over time is likely to help matters. Furthermore, it is not necessary to introduce a new paradigm in order to provide the expressive power required to validate data: see [Patel15] for a discussion on the applicability of description logics to constraint checking.

Theoretical studies on the unification of RDF and OWL

The challenges for reasoning caused by the unification of RDF and OWL have been extensively discussed in the literature. For example, even before OWL was standardised, Patel-Schneider and Fensel [Patel02] had succinctly summarised the issue. "[D]efining the model theory of OWL as an extension of the model theory of RDF and representing OWL constructs syntactically in RDF leads to paradoxical situations, i.e., ill-defined model theories for OWL". The author's presented five approaches to layering OWL over RDF. This included a clear identification of the presence of Russell-type paradox when OWL is directly layered over RDFS as a same-syntax solution. Nonetheless this was the approach adopted in OWL-Full. The ramifications of these decisions live on today - our challenge 1 for validating linked data schemata.

Later, when the OWL standard was agreed, the principal authors of the OWL Semantics documented the "difficult trade-offs" that they made during the design of OWL version 1 [Hor03]. Horrocks et al. identify four broad classes of problems: syntactic, semantic, expressive power and computational problems. While the latter two categories seem less relevant for our current work it was in fact the computational overheads introduced by allowing the use of classes as instances that led to the exclusion of this feature from OWL-DL. As will be seen in our experimental work, under the impredicativity heading, this causes many issues in observed linked data. In section 6.2 of that paper, it is shown how the solution proposed for dealing with malformed OWL syntax expressed in RDF leads to the creation of additional anonymous classes despite them being "almost certainly not what was intended by the user". This is an example of our challenge 4 (Permissivity of OWL) that leads to standard reasoners being unable to detect these issues in linked data schemata without human inspection of the resultant reasoned ontology. In contrast, our approach highlights these potential errors and presents them to the user or validator for verification. In their discussion on future extensions, while being aware that the import of ontologies by others was likely to become the norm in the semantic web (as we now see in linked data), they described the OWL import facility as "very trivial". This is the source of our challenges 2 and 3 for linked data schemata validation. They also highlighted the fact that the closed world and unique name assumptions are useful in some situations, despite them being outside the scope of the general OWL model. Our advances in schemata validation, and the experimental evidence we have collected, confirm the practical utility of reasoning within closed world and uniquely named regimes, for validation, even on the open web.

When the OWL standard was revised as OWL2 in 2008, the process was again documented by some of the principal authors [Grau08]. Grau et al. identify additional issues with OWL1 ontology specifications including an additional complication in the import semantics whereby two (or more) ontologies written in the same OWL species can interact in unpredictable and unintuitive ways when one ontology imports the other, leading to a new ontology that is contained in a different species. OWL 2 introduces the idea of declaration consistency which means all types must be declared, although not syntactically necessary, this allows some additional validation checks to catch mistyping of entity terms. However this approach is not applicable to linked data that is not written in OWL 2 and hence the validation checks performed by our approach can detect these trivial typing

errors e.g. misspelt names of classes, in a much broader range of data. OWL2 also improves support for imports by tightening the specification of ontology name URIs as both name and published location of the ontology on the web, however as our experimental results show (see later) not all widely used and imported ontologies in the current web of data can be discovered at their stated locations. Support for ontology versioning management has also been added, but this is still primitive and the species or profile impacts of imports is still unintuitive and unpredictable.

Most recently, as the success of linked data, with billions of triples published for public consumption, has become apparent, the question of data quality and hence validation has come to the fore [Mendes12][Bizer07]. One barrier to progress has been the strong association between OWL and open world reasoning, which has limited the exploration of constraint validation using description logics under closed world assumptions. Patel-Schneider [Patel15] has discussed the issues and approaches when applying description logic to validation, and attacked the perception that the closed world and unique name interpretations have no place in the description logic world – to which OWL belongs. Although Patel-Schneider's technical approach to linked data validation, based on RDFS and SPARQL , differs from our custom OWL reasoner-based approach, we follow the same direction with respect to selectively applying closed world and unique name assumptions in validation.

Reasoning and consuming linked data research

The original semantic web vision included the goal of applying reasoning at a web scale [TBL01]. Linked data provides the basis of the current web of data and so reasoning over it has naturally been tackled by several researchers, see for example [Kiryakov09], [Demartini12] and [Polleres13]. Given the divergence of linked data from the semantic web ideal, a wide variety of non-standard and unanticipated reasoning approaches have been developed - from probabilistic techniques [Demartini12] to rule-based approaches [Kiryakov09]. Given our interest in RDFS and OWL-based schemata validation, we focus here on approaches that support the RDFS and OWL standards. The challenges for applying reasoning to linked data, were laid out by Polleres et al. as data scale, data heterogeneity (mixing of OWL DL, OWL Full and RDFS), data inconsistency, data dynamics and extending inference beyond RDFS and OWL. Data scale is less of an issue for our work since we focus on schemata and thus primarily "TBox" assertions, although as our experimental work shows, some linked data schemata are themselves very large, even without considering instance data. Nonetheless, the problems of operating on billions of triples, the focus of much active research in open web reasoning, are out of scope for our validator. Tackling heterogeneity is a focus of our work (challenge 1), but whereas we aim to identify inconsistent or incomplete schemata in order to fix them and improve quality, the predominant approach towards reasoning within the linked data community is to try and do the best possible with the triples available. While appropriate for many use cases, it often requires strategies that weaken consistency or soundness constraints to make the problem tractable, silently discard problematic data or conservatively reduce the materialisation of inferred triples compared to completely applying the OWL Direct Semantics [Polleres13]. Although there are points of similarity, in general these approaches produce weaker validation results than our approach since they are less complete and not sound.

4. Linked Data Schemata Validation in the Dacura Quality Service

In order to meet the challenges of linked data schemata validation, we have developed the Dacura Quality Service (DQS) and the Dacura Schema Manager. Both are integrated into our Dacura platform for data curation, which has been introduced elsewhere [Feeney14]. The Dacura Schema Manager (fig.1) provides a user interface for loading new linked data schemata into the system. It identifies and recursively loads all the implicitly or explicitly imported vocabularies or ontologies

from the web, creates a master schema based on the union of all referenced terms, gathers statistics regarding the strength of the dependencies between ontologies, identifies incidences of ontology hijacking and performs analyses the distribution of definitions within the ontologies to highlight likely areas of poor specification (e.g. term definitions being distributed across multiple parts of the vocabulary). The Quality Test screen of the Dacura Schema Manager then allows the user to select specific reasoner-based validation checks, call the DQS through its API to perform the checks and render the results in human-readable form.

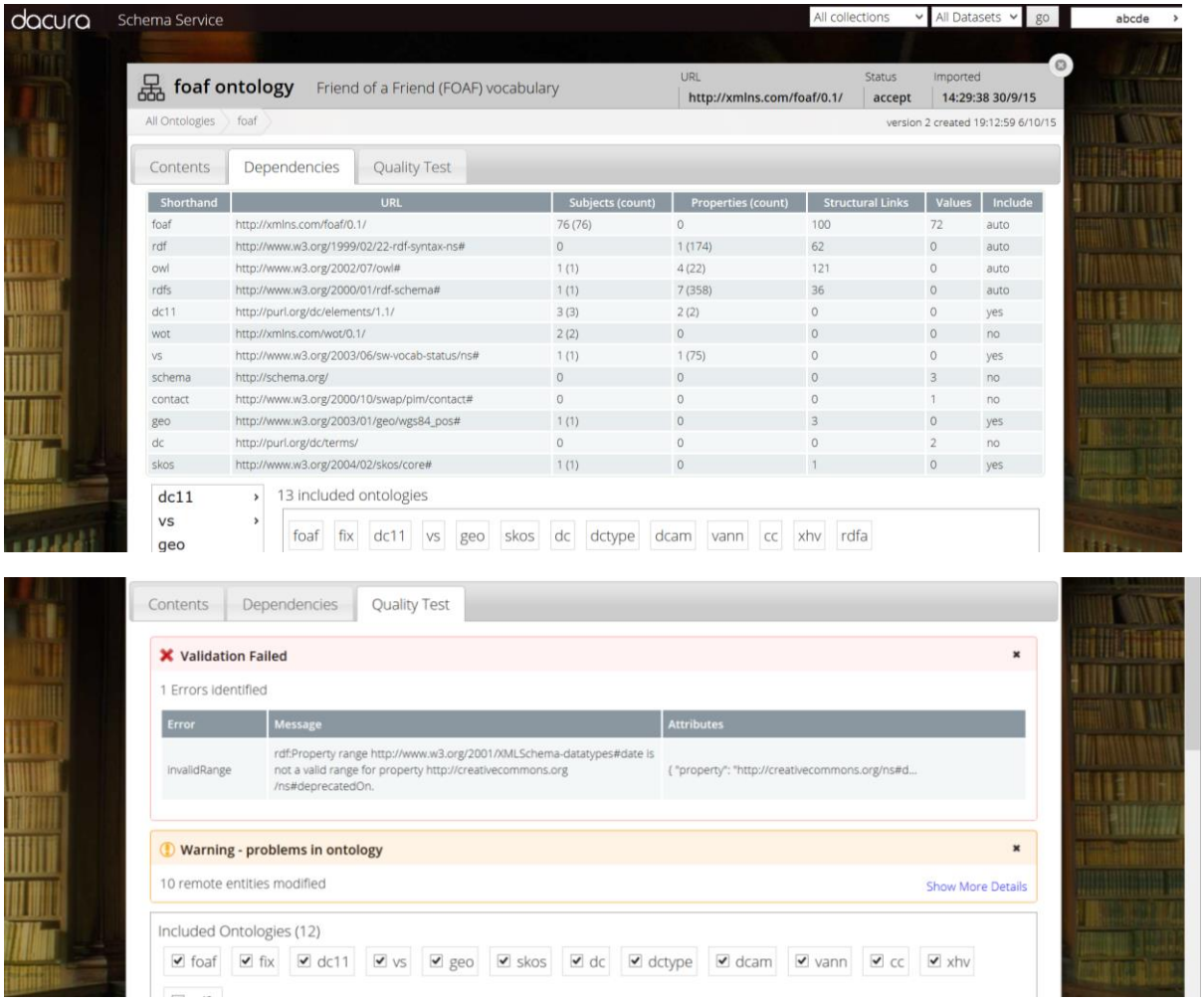

Figure 1: Dacura Schema Manager: screenshots of dependency analysis (above) and DQS (below)

The DQS provides an ACID (Atomic, Consistent, Isolated, Durable) triplestore for the storage of OWL ontologies and instance data, and a gatekeeper function which ensures the consistency of its stored data. We treat consistency as both internal consistency of the OWL ontology and consistency of instance data with respect to this ontology. In this way the DQS provides a triplestore in which stored information always respects the ontology: updates which are not consistent are not allowed. If the schema changes, the instance data is required to also change in a fashion conformant to the new schema. The DQS is built in SWI Prolog in the ClioPatra semantic web infrastructure [Wielemaker15]. The source code is available online under a GPL2 license[3].

[3] Source code available at https://github.com/GavinMendelGleason/dacura

To do this, we have built a custom reasoner as part of the DQS which interprets ontologies under a relatively large but custom fragment of OWL DL (see table 1 for the OWL 2 features implemented so far) subject to additional constraints that increase the ability of the reasoner to deal with the unification of OWL and RDF/RDFS in linked data schemata (challenge 1, challenge 2), detect likely validation errors (challenge 3, challenge 4) and improve efficiency. This fragment of OWL DL has also been shaped by the modelling requirements of ontology development for the Seshat:Global History Databank [Turchin15] which is our initial use case, as well as the OWL 2 terms we found most often used in linked data schemata on the web of data. The range of support for OWL 2 constructs has substantially increased from our earlier paper [MG15] which focused on RDFS. It is anticipated that we will continue to extend the support for further OWL 2 features in future work.

The overall strategy of the DQS reasoner is not to prove that there is a possible model for any given ontology but instead to reject those ontologies that either cannot have a possible model, or which are incompletely specified and require the inferring of new classes to form a model (as, in practice, these are often caused by user errors) under a closed world assumption. As the Dacura Schema Manager service imports vocabularies for the reasoner as they are refered to (wherever possible), the closed world in our case corresponds to the whole of the web of data, at the level of schema specification. We do not claim that the reasoner is sound or complete under OWL DL, just that it is capable of detecting many errors in linked data schemata, including errors undetectable under standard reasoning. Our approach is supported by building a subsumption prover in SWI Prolog. Due to the difficulty of performing subsumption computations with equivalences, we have opted in DQS to ignore non definitional equivalence and require that there are no cycles in the declared subsumption of classes or predicates. This again does not give us the full power of OWL DL, however, in our experimental evaluations, we have still yet to find any evidence for the intentional use of class or property cycles, and our trade-off thus appears likely to have little cost in practice.

Table 1 OWL 2 terms and features supported by DQS Reasoner

Language Elements	Supported	**Language Elements (cont.)**	Supported	**Axioms and Assertions (cont.)**	Supported
Classes, Datatype and Restriction		owl:hasValue	Y	**Property Expression Axioms**	
owl:Class	Y	owl:SelfRestriction	N	rdfs:subPropertyOf	Y
owl:intersectionOf	Y	**Special classes**		owl:inverseOf	Y
owl:unionOf	Y	owl:Thing	Y	owl:equivalentProperty	N
owl:complementOf	Y	owl:Nothing	Y	owl:property DisjointWith	Y
owl:oneOf	Y	**Properties**	Y	rdfs:domain	Y
rdfs:Datatype	Y	owl:DatatypeProperty	Y	rdfs:range	Y
owl:datatypeComplementOf	N	owl:ObjectProperty	Y	owl:propertyChain	Y
owl:oneOf	Y	**Special properties**		owl:FunctionalProperty	Y
owl:onDatatype	Y	owl:TopDataProperty	Y	owl:InverseFunctionalProperty	N
owl:withRestrictions	Y	owl:BottomDataProperty	Y	owl:ReflexiveProperty	N
owl:Restriction	Y	owl:TopObjectProperty	Y	owl:IrreflexiveProperty	N
owl:onProperty	Y	owl:BottomObjectProperty	Y	owl:SymmetricProperty	N
owl:onClass	Y	**Individuals**		owl:AsymmetricProperty	N
owl:onDataRange	Y	owl:NamedIndividual	N	owl:TransitiveProperty	Y
owl:onProperties	Y	**Axioms and Assertions**		owl:hasKey	Y
owl:cardinality	Y	**Class Expression Axioms**		**Assertions**	
owl:maxCardinality	Y	rdfs:subClassOf	Y	**owl:NegativePropertyAssertion**	N
owl:minCardinality	Y	owl:equivalentClass	N	owl:sourceIndividual	N
owl:minQualifiedCardinality	Y	owl:disjointWith	N	owl:assertionProperty	N
owl:minQualifiedCardinality	Y	owl:disjointUnionOf	Y	owl:targetValue	N
owl:qualifiedCardinality	Y	**Individual Axioms**		owl:targetIndividual	N
owl:allValuesFrom	Y	owl:differentFrom	N	owl:AllDifferent	N
owl:someValuesFrom	Y	owl:sameAs	N	owl:AllDisjointClasses	N
				owl:AllDisjointProperties	N
				owl:members	N

DQS provides an interface to a triple store via HTTP using a simple JSON format for updates (both inserts and deletes) of triples, and of both instance and ontology data. The service responds to updates either with a success message stating that the insertion is consistent, or a message describing the precise reason for failure of consistency according to the reasoner. The reasoner takes pains to ensure that it builds up a witness of failure which demonstrates the counter-example to consistency satisfaction which can then be used by the client to come up with a suitable strategy for dealing with the failure. The results of our evaluation of linked data schemata (see sections 5 and 6) were compiled by loading ontologies in the Dacura Schema Manager, and then testing them against the Dacura Quality Service, and then looking at the error reports provided. Next we examine the specific solutions implemented in the Dacura Schema Manager and Dacura Quality Service to address the challenges of linked data schemata validation.

Overcoming Challenge 1 (Heterogeneity)

As discussed in the background section, the free mixing of RDF, RDFS and OWL triples gives rise to different interpretations. Our approach is to deliberately misinterpret as OWL the RDF/RDFS classes and properties that are normally outside the scope of OWL-DL, when there is no immediate conflict in doing so, e.g. we treat a rdfs:class as equivalent to an owl:class. This interpretation doesn't present an insurmountable difficulty for reasoning. Similarly rdf:Property is treated at an equivalent level to owl:DatatypeProperty and owl:ObjectProperty and no overlap is allowed between them. All domains and ranges that are asserted are checked to ensure they support subsumption. Misuse of language features and low level RDF syntax with reserved meaning in OWL such as rdf:List is also detected as an error.

This approach is applicable in situations where data is going to be published according to the combined ontology, or internally within a system which interprets the instance data as OWL. This is in line with common practice for linked data but presents potential problems for interoperability of the produced linked data since OWL reasoners might deem it inconsistent due to the fact that we still allow a mix RDFS and OWL and hence are not a proper subset of OWL DL. In any case, having completed the semantic integration, only a synctactic transformation would be required to render our models fully OWL compliant. As our experimental work will show, dealing with the commonly used vocabularies on the web of data today requires handling a free mix of RDFS and OWL terms.

Overcoming Challenge 2 (Imports)

Since there are a range of ways that linked data schemata reference or import each other it was necessary to define a mechanism to construct the composite ontology defined by a linked data schemata to enable validation under a closed world assumption. For this reason, we have treated all dependencies to external namespaces as implicit owl:imports.

Dependencies between ontologies were defined as either property dependence or structural dependence:

- Property dependence: if an ontology A uses a property from another ontology B, then A is considered to have a property dependence on B.
- Structural dependence: if an ontology A contains a statement which defines its classes or properties in terms of entities in ontology B, then A is considered to have a structural dependence on B. Table 2 shows the specific OWL terms which we treat as creating structural links between ontologies.

Other references to external URIs in a schema were ignored and not treated as dependencies.

Table 2 – OWL/RDF/RDFS terms that create dependencies between ontologies

Namespace	Term
rdf	type
rdfs	range, domain, subPropertyOf, subClassOf, member
owl	inverseOf, unionOf, complementOf, datatypeComplementOf, intersectionOf, oneOf, dataRange, disjointWith, imports, allValuesFrom, someValuesFrom, equivalentClass, equivalentProperty, disjointUnionOf, propertyDisjointWith, members, disjointWith, propertyDisjointWith, onProperty, onClass, propertyChainAxiom

The Dacura Schema Manager tool implements these rules to analyse any given ontology and recursively create its dependency tree, fetch the constituent ontologies or vocabularies and create a union between them for checking by the DQS.

Overcoming Challenge 3 (Distributed Authoring)

The Dacura Schema Manager detects all dependencies between ontologies as described in the last section. This forms the basis for detecting references to missing or unavailable ontologies. Similarly it can detect namespace violations in input ontologies such as ontology hijacking. The logical consequences of building unified models from many ontologies are detected by the DQS, it especially highlights situations where local work-arounds have been made that render the unified model inconsistent.

Overcoming Challenge 4 (OWL Permissivity)

By applying the closed world assumption to the full graph specifying a linked data schema imported from the web of data, it is possible to detect orphan classes. These are rejected by DQS as incompletely specified (similar to the use of declarations in OWL 2 but without the need to augment existing ontologies with these new declarations). In addition, the detection of subsumption failures and cycles in class or property declarations allows us to detect further potential misuse of OWL features.

5. Evaluation Methodology

We started our evaluation of the interoperability of the various ontologies and vocabularies commonly used by linked data documents by choosing a reasonably representative linked data ontology, then identifying the set of ontologies and vocabularies that it depended on and including these ontologies, then identifying the set of ontologies needed by these included ontologies, including them and continuing until all of the dependencies had been included or were deemed to be impossible to include. This produced a breadth-first dependency tree for the target ontology (figure 2).

Our initial target ontology was the Open Annotation ontology from the draft 2 specification [OA13]. This ontology was selected for both practical and theoretical reasons. Practically, we wished to implement a system for the Seshat: Global History Databank in which users could annotate content at a variety of scopes and we wanted to be able to validate instance data which was expressed according to the ontology and we identified the Open Annotation model as the most promising work in the space. Theoretically, it represented a good example of a linked data schema in the wild – it has been constructed by a W3C community group according to the linked data principles, using well known third party vocabularies and ontologies, and it is in use in practice.

Identifying Dependencies

We applied Dacura Schema Manager's dependencies tool to the open annotation ontology. Figure 2 shows the dependency tree that it produced and Table 3 shows the 62 specific ontologies that were included. We then analysed each of these ontologies with the DQS tool to identify to what extent they exhibited problems in terms of creating a unified knowledge model that incorporated them. It should be noted that the ontologies that were included through this dependency analysis almost all stem from the inclusion of the Dublin Core suite of vocabularies – which are easily the most commonly used vocabularies in the linked data world (as shown by the dependency matrix that we constructed). Therefore, most of the dependency tree shown here is common to almost all linked data vocabularies.

Table 3. Ontologies analysed as part of this work.

shorthand	URL	Description
adms	http://www.w3.org/ns/adms#	Asset Description Metadata Schema (ADMS)
ao	http://purl.org/ontology/ao/core#	The Association Ontology
atom	http://bblfish.net/work/atom-owl/2006-06-06/#	Atom syndication format
bibo	http://purl.org/ontology/bibo/	The Bibliographic Ontology
bio	http://purl.org/vocab/bio/0.1/	BIO: A vocabulary for biographical information
cc	http://creativecommons.org/ns#	Creative Commons
contact	http://www.w3.org/2000/10/swap/pim/contact#	Contact: Utility concepts for everyday life
crm	http://purl.org/NET/cidoc-crm/core#	CIDOC Conceptual Reference Model
dbox	http://dublincore.org/documents/dcmi-box/	(Empty) DCMI-Box encoding scheme
dc	http://purl.org/dc/terms/	DCMI Metadata Terms - other
dc11	http://purl.org/dc/elements/1.1/	Dublin Core Metadata Element Set, Version 1.1
dcam	http://purl.org/dc/dcam/	Metadata terms related to the DCMI Abstract Model
dcat	http://www.w3.org/ns/dcat#	The data catalog vocabulary
dctype	http://purl.org/dc/dcmitype/	DCMI Type Vocabulary
doap	http://usefulinc.com/ns/doap#	Description of a Project (DOAP) vocabulary
doc	http://www.w3.org/2000/10/swap/pim/doc#	Document vocabulary
dtest	http://www.w3.org/2006/03/test-description#	Test Description Vocabulary
dul	http://www.loa-cnr.it/ontologies/DUL.owl#	DOLCE+DnS Ultralite
event	http://purl.org/NET/c4dm/event.owl#	The Event ontology
foaf	http://xmlns.com/foaf/0.1/	Friend of a Friend (FOAF) vocabulary
frbr	http://purl.org/vocab/frbr/core#	Expression of Core FRBR Concepts in RDF
geo	http://www.w3.org/2003/01/geo/wgs84_pos#	WGS84 Geo Positioning
geometry	http://data.ordnancesurvey.co.uk/ontology/geometry/	A ontology to describe abstract geometries.
gn	http://www.geonames.org/ontology#	The Geonames ontology
grddl	http://www.w3.org/2003/g/data-view#	GRDDL Gleaning Resource Descriptions
hcard	http://purl.org/uF/hCard/terms/	HCard Vocabulary
http	http://www.w3.org/2006/http#	A namespace for describing HTTP messages

iana	http://www.iana.org/assignments/relation/	Link Relations
ical	http://www.w3.org/2002/12/cal/ical#	RDF Calendar
icalspec	http://www.w3.org/2002/12/cal/icalSpec#	ICAL specifications
keys	http://purl.org/NET/c4dm/keys.owl#	Musical keys
label	http://purl.org/net/vocab/2004/03/label#	Term definitions for singular and plural label properties
leo	http://linkedevents.org/ontology/	LODE: An ontology for Linking Open Descriptions of Events
log	http://www.w3.org/2000/10/swap/log#	Logic Ongology
mo	http://purl.org/ontology/mo/	The Music Ontology
neogeo	http://geovocab.org/spatial#	A vocabulary for describing topological relations between features
oa	http://www.w3.org/ns/oa#	Open Annotation Data Model
ont	http://www.w3.org/2006/gen/ont#	An Ontology for Relating Generic and Specific Information Resources
opmv	http://purl.org/net/opmv/ns#	The Core OPMV Vocabulary
ov	http://open.vocab.org/terms/	Open Vocabulary
prov	http://www.w3.org/ns/prov#	W3C PROVenance Interchange Ontology (PROV-O)
qb	http://purl.org/linked-data/cube#	The data cube vocabulary
rdfa	http://www.w3.org/ns/rdfa#	RDFA specification
rdfg	http://www.w3.org/2004/03/trix/rdfg-1/	RDF Graph
rel	http://purl.org/vocab/relationship/	A vocabulary for describing relationships between people
rev	http://purl.org/stuff/rev#	RDF Review Vocabulary
schema	http://schema.org/	Schema.org (converted to OWL by TopQuadrant)
scovo	http://purl.org/NET/scovo#	The Statistical Core Vocabulary (SCOVO)
sim	http://purl.org/ontology/similarity/	The Similarity Ontology
sioc	http://rdfs.org/sioc/ns#	Semantically Interlinked Online Communities - core
sioctypes	http://rdfs.org/sioc/types#	SIOC Types Ontology
skos	http://www.w3.org/2004/02/skos/core#	SKOS Vocabulary
time	http://www.w3.org/2006/time#	An OWL Ontology of Time (OWL-Time)
timezone	http://www.w3.org/2006/timezone#	A time zone ontology
ubench	http://swat.cse.lehigh.edu/onto/univ-bench.owl#	An university ontology for benchmark tests
vann	http://purl.org/vocab/vann/	VANN: A vocabulary for annotating vocabulary descriptions
vcard	http://www.w3.org/2006/vcard/ns#	Vcard vocabulary
voaf	http://purl.org/vocommons/voaf#	Vocabulary of a Friend
void	http://rdfs.org/ns/void#	Vocabulary of Interlinked Datasets (VoID)
vs	http://www.w3.org/2003/06/sw-vocab-status/ns#	SemWeb Vocab Status ontology
wdrs	http://www.w3.org/2007/05/powder-s#	POWDER-S Vocabulary
xhv	http://www.w3.org/1999/xhtml/vocab#	XHTML specification

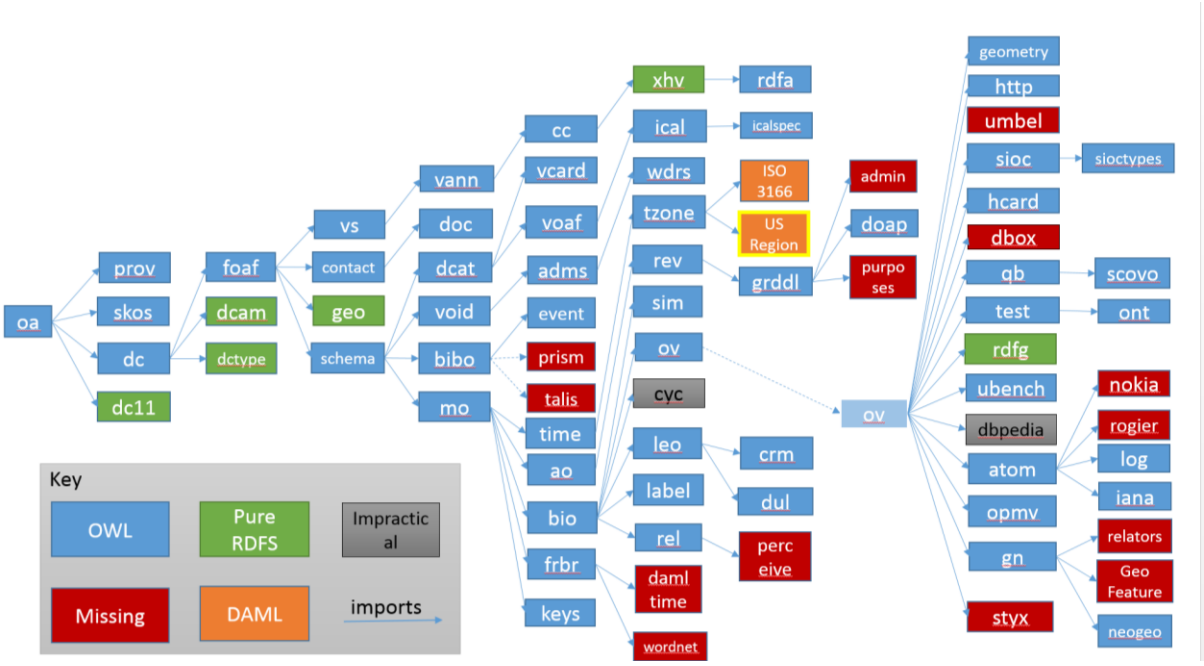

Figure 2 Open Annotation dependency tree of linked data vocabularies and ontologies

Our dependency analysis terminated whenever we came to an ontology that we could not retrieve, either because we discovered that the ontology no longer existed (e.g. WordNet), or because we proved unable to locate a machine-readable version of the ontology on the internet. In two cases our dependency tree brought us to ontologies that were simply too big for our tools to handle – OpenCyc and DBpedia. There was only a single structural link to the DBpedia ontology from the rest and 2 links to OpenCyc, so these ignored ontologies were not deeply integrated into the global linked data schema model that we constructed. We can assume, however, due to the nature of these models, that they would have greatly increased the number of dependant ontologies. In two cases, dependant ontologies were written in DAML, a predecessor of OWL and these ontologies were not automatically analysed as our tools were not capable of interpreting them. Manual analysis of both revealed that they had no further dependencies.

Schema Validation

Once the dependency tree of ontologies for Open Annotation had been established, the composite schema so defined (consisting of the union of all of the constituent ontologies) was analysed by the DQS reasoner for validation errors. See the next section for the results.

6. Validation Results

Our analysis of the 62 ontologies revealed that 25 ontologies contained namespace violations (ontology hijacking: making assertions about entities defined in other ontologies with global scope). 8 ontologies contained dependencies on a total of 11 missing ontologies. 3 ontologies contained basic errors that were categorised as typos. 12 ontologies contained statements that are illegal in OWL DL due to them being impredicative – predicating over classes or properties which is illegal in first order logic entirely - and basic misuses of language constructs (e.g subclassing owl:differentFrom and expecting its semantics to be retained). One ontology contained both property and class cycles – and in both cases manual analysis of the ontologies revealed that they were, as anticipated, likely to have been the result of specification errors rather than obtuse ways of defining a single class or property. The validation results are presented in the tables below.

References to Missing Ontologies

As is to be expected in the evolving web of data a number of the referenced ontologies were no longer (or currently) unavailable. However as a community we must be aware of the implication of this for linked data quality –if schema specifications are going to be rendered incomplete due to changes in the availability of imported ontologies or terms then it places a limit on the degree of validation we can perform.

Table 4: References to missing or unavailable dependencies detected

Ontology	Missing (or unavailable) Dependencies
atom	property: http://eulersharp.sourceforge.net/2004/04test/rogier#productProperty (1 use) atom:Content rdfs:subClassOf http://sw.nokia.com/WebArch-1/Representation atom:Content rdfs:subClassOf http://sw.nokia.com/WebArch-1/Representation rdfs:subPropertyOf http://sw.nokia.com/WebArch-1/representation atom:scheme rdfs:range http://sw.nokia.com/WebArch-1/InformationResource atom:src rdfs:range http://sw.nokia.com/WebArch-1/InformationResource _:atom31 rdf:type file:///Users/hjs/Programming/sommer/www/atom/2006-06-06/AtomOwl.n3#update _:atom37 rdf:type file:///Users/hjs/Programming/sommer/www/atom/2006-06-06/AtomOwl.n3#rel
doap	doap:Project rdfs:subClassOf http://xmlns.com/wordnet/1.6/Project (1 use)
frbr	frbr:Work rdfs:subClassOf http://xmlns.com/wordnet/1.6/Work~2 frbr:Event rdfs:subClassOf http://www.isi.edu/~pan/damltime/time-entry.owl#Event
gn	gn:Feature owl:equivalentClass http://www.mindswap.org/2003/owl/geo/geoFeatures20040307.owl#GeographicFeature
grddl	Properties: http://www.rddl.org/purposes#normative-reference (3 uses) http://webns.net/mvcb/generatorAgent (1 use)
neogeo	http://geovocab.org/spatial owl:imports http://geovocab.org/mappings/spatial
timezone	timezone owl:imports http://www.daml.org/2001/09/countries/iso-3166-ont timezone owl:imports http://www.daml.ri.cmu.edu/ont/USRegionState.daml
dbox	Ontology is empty – contains no classes

Ontology Hijacking / Namespace Violations

A widespread pattern observed in the ontologies under analysis was the presence of assertions designed to support interoperability of ontologies. For example, a very common pattern observed was to specify that certain properties from imported ontologies were defined to be of type owl:AnnotationProperty – in order to allow them to be processed by standard OWL tools which do not know how to deal with properties defined as rdf:Property. The basic problem with this pattern is that it amounts to non-coordinated interoperability on an individual library scope – each ontology attempts to handle interoperability for its own scope, but when these ontologies are combined together, each piecemeal attempt at interoperability is combined into a common model and the union of these piecemeal attempts at library level interoperability without any facilities for modularity leads to a mess.

Table 5: Namespace violations detected

Ontology	Count	Third party ontologies altered (number of entities altered)
Atom	5	iana
Bibo	50	rdf (3), rdfs (1), owl (2), dc (19), skos (6), vs (1), event (7), foaf (11)
Crm	10	vann (2), dc (3), cc (1), label (1), skos (3)
event	8	dc11 (3), foaf (3), geo (1), vs (1)
Foaf	10	owl (1), rdfs (1), dc11 (3), vs (1), geo (1), skos (1), wot (2 – only use)

frbr	32	rdf (1), foaf (3), dc (5), dc11 (7), vann (3), skos (1), cc (11), geo (1)
geometry	3	rdfs (2), dc11 (1)
gn	3	foaf (1), skos (2)
grddl	1	owl
http	2	rdfs (1), xsd (1)
icalspec	2	xsd
leo	4	crm (3), event (1)
mo	1	vs
opmv	6	owl (1), time (5)
prov	6	owl (2), rdfs (4)
rel	1	foaf
rev	9	rdfs (2), dc11 (3), foaf (2), vs (2)
sim	5	owl (1), dc (2), vs (1), foaf (1)
sioc	10	dc (5), foaf (5)
sioctypes	2	skos (1), sioc (1)
time	1	timezone

The second major category of namespace violations observed in the data are illegal assertions that serve to silently kill error reporting in tools. For example the assertion: rdfs:Class a owl:Class is used in two separate ontologies – it declares that an RDFS class is an instance of an OWL class[4] – an interpretation that is not true under either OWL DL or Full but it manages to successfully silence error checking in a number of tools. These type of assertions are particularly problematic in distributed ontologies because they can make the knowledge model silently inconsistent. They also break robustness principles by deliberately producing malformed specifications rather than compensating for real-world variation and noise at input.

Finally, in a certain number of cases, ontologies knowingly and explicitly change other ontologies for convenience in utilising external class definitions. This type of usage is most pointedly described in the bibo ontology:

dc:Agent a owl:Class ; owl:equivalentClass foaf:Agent ;

An editorial note in the ontology states: “BIBO assert that a dcterms:Agent is an equivalent class to foaf:Agent. This means that all the individuals belonging to the foaf:Agent class also belongs to the dcterms:Agent class. This way, dcterms:contributor can be used on foaf:Person, foaf:Organization, foaf:Agent and foaf:Group. Even if this link is not done in neither the FOAF nor the DCTERMS ontologies this is a wide spread fact that is asserted by BIBO.”

In such cases it would be more appropriate to use local sub-classing to achieve the equivalent effect without over-writing the definitions in external namespaces.

Typos

Three ontologies were found to contain basic errors which were interpreted as typos – using xsd:int (which does not exist) instead of xsd:integer; using rdfs:ramge; and using the wrong namespace for xsd. The presence of such errors in long established and public ontologies, highlights the lack of tool support for ontology validation – they are obvious errors but they have not been identified by standard OWL reasoners.

[4] Note this is different from interpreting rdfs:Class and owl:Class as being indentical constructs, as we do, this patch declares rdfs:Class as an owl:Class, like any other owl:Class.

Table 6: Typos detected

Ontology	Typos (underlined)
contact	contact:assistant rdfs:ramge foaf:Agent contact:participant rdfs:ramge foaf:Agent
qb	qb:order rdfs:range xsd:int
cc	cc:deprecatedOn rdfs:range <http://www.w3.org/2001/XMLSchema-datatypes#date> .
atom	atom:length rdfs:range xsd:int

Impredicativity / misuse of language constructs

Since OWL DL is a first order theory, it is not possible to quantify over classes and predicates. Yet no such restriction exists in RDF. This leads to a number of problems when using OWL ontologies which reference RDF ontologies which make use of higher-order and impredicative features.

In the very widely used **dc** ontology, the rdf:type relation is defined with the range of rdfs:Class. This is immediately problematic as rdfs:Class is the class of all classes and such impredicative statements are illegal in OWL DL and are dangerous regardless, due to the very real threat of paradox. Similarly **dc** uses the rdf:subClassOf relation to derive a subclass of the class of classes. This again is higher order reasoning, without any guarantee of predicativity.

In **skos** we see the use of rdf:List as a range, but rdf:List is an internal syntactic element of OWL. Free mixing of rdf:first and rdf:next would leave reasoners unable to distinguish between what is intended as a property and what is intended to be syntax of the language itself. While this problem has been described thoroughly [Horr03], it also has not been stamped out in the wild, and **skos** is a very widely used ontology purporting itself to be OWL.

In **gn, log, void, qb, wdrs, atom** and **voaf** we see the very common use of higher order logic, with subclassing of class, properties, and assignation of ranges over properties and classes. In most of these cases, manual analysis reveals that the statements were probably unnecessary. However, this is not to say that higher order reasoning cannot sometimes be useful. We will discuss later how such things can be achieved without stepping into undecidability.

In **atom** there is an even more unusual metalogical statement, making a statement about statements themselves! Without some sort of stratification, such logic is risky at best. **Atom** additionally makes use of inference facilities that are not themselves part of OWL. Utilising ontologies of this form requires a tool chain which is capable of making these inferences, something that is not widely deployed and available.

Table 7: Instances of impredicativity/misuse of language constructs detected

Ontology	Triple(s)	Error Description
dc	dc:type rdfs:range rdfs:Class ;	Predicating over class
dc	dc:AgentClass rdfs:subClassOf rdfs:Class	Overriding basic language construct
skos	skos:memberList rdfs:range rdf:List ;	rdf:List is an internal structural element of OWL – it can't be used directly
grddl	grddl:TransformationProperty rdfs:subClassOf owl:FunctionalProperty ;	Higher order use of the rdfs:subClassOf relation
wdrs	wdrs:Document rdfs:subClassOf owl:Ontology .	Higher order use of the rdfs:subClassOf relation
rel	rel:friendOf rdfs:subPropertyOf owl:differentFrom (32 times)	Higher order use of the rdfs:subClassOf relation

atom	atom:RelationType rdfs:subClassOf owl:ObjectProperty .	Higher order use of the rdfs:subClassOf relation
atom	atom:Link rdfs:subClassOf rdf:Statement	Creating subclasses of an already higher order feature
atom	atom:rel rdfs:subPropertyOf rdf:predicate	Creating subclasses of an already higher order feature
atom	atom:subject rdfs:subPropertyOf rdf:subject	Creating subclasses of an already higher order feature
atom	atom:to rdfs:subPropertyOf rdf:object	Creating subclasses of an already higher order feature
bio	bio:differentFrom rdfs:subPropertyOf owl:differentFrom (15 times)	Higher order use of the rdfs:subClassOf relation
gn	gn:featureClass rdfs:subPropertyOf dc:type ;	Using impredicative property from dc
gn	gn:featureCode rdfs:subPropertyOf dc:type	Using impredicative property from dc
log	log:definitiveDocument rdfs:domain rdf:Property	Predicating over class of properties
Log	log:definitiveService rdfs:domain rdf:Property ;	Predicating over class of properties
Void	void:linkPredicate rdfs:range rdf:Property	Predicating over class of properties
Void	void:property rdfs:range rdf:Property	Predicating over class of properties
Voaf	voaf:occurrences a owl:objectProperty, rdfs:range xsd:integer	Mismatch between objectProperty and literal range type
Qb	qb:parentChildProperty rdfs:range rdf:Property	Predicating over class of properties
Qb	qb:ComponentProperty rdfs:subClassOf rdf:Property	Higher order use of the rdfs:subClassOf relation

Property / Class Cycles

Table 8 presents the class or property cycles detected in the **crm** ontology. The first example, asserts that a legal body is equivalent to a group, which seems highly questionable, though it would require the **crm** authors to confirm. The second looks more likely, but still questionable, where they establish an equivalence between “bearing a feature” and “being composed of”.

We have also noticed that many statements of equivalence were between classes in different ontologies, establishing a link between an element in one place, and that in another. However, these equivalences were often coupled with *additional* qualifications. Such behaviour completely negates the capacity to use linked data in an interoperable fashion, as the original publisher of the ontology may very well have instance data which is deemed invalid when read by the second publisher, and vice versa. This “ontology hijacking” [Hogan10] should be highly discouraged.

Table 8: Property/Class cycles detected in crm ontology

Triple(s)	**Problem**
crm:E40_Legal_Body rdfs:subClassOf crm:E74_Group crm:E74_Group rdfs:subClassOf ns1:E40_Legal_body	Cycle in class hierarchy
crm:P46_is_composed_of rdfs:subPropertyOf crm:P56_bears_feature; crm:P56_bears_feature rdfs:subPropertyOf crm:P46_is_composed_of	Cycle in property hierarchy

7. Recommendations for Correcting Problems in Linked Data Schemata

Given our experiences in constructing the DQS and the experimental analysis performed of real world linked data schemata, we offer the following recommendations for improving best practice in linked data vocabulary design.

Metareasoning with rdf:List

Ontologies in OWL need to immediately stop using the underlying syntactic elements of rdf:List within the logic of the ontology, as is done in SKOS. The appropriate way to deal with this problem is to have a drop in replacement for RDF collections written in OWL such that there is no syntactic/logical mixing. There have been some list ontologies constructed such as the Ordered List Ontology[5] , CO-ODE List Ontology[6], however what is needed is a drop in replacement for RDF collections in general. Bags are trivial to construct in OWL, and both ordered lists and indexed sequences have also been demonstrated, so creating such an ontology is more a collation task than an ontology engineering one. Migrating current OWL ontologies to use such a drop in replacement would be a relatively minor task and would allow them to be compliant OWL DL.

Impredication and Higher order features

The impredicative and higher order features of RDF are used in a fair number of ontologies, albeit in a relatively minor way. Supporting such behaviour does not require abandoning soundness or allowing paradox. Type theory, going back to Russell, has developed techniques to avoid impredicative paradoxes through the use of some form of stratification, which could be used to extend OWL DL. The complexity or indeed decidability of such an extension remains to be explored.

Piecemeal Inter-operability

The motivation for ontology publishers to include assertions that are designed to make it easier for tool users to interact with their vocabularies is understandable, altruistic and undoubtedly helpful in practice when the problems that users often have when interacting with semantic web technologies are taken into account. However, when published as part of the body of an ontology, such language interoperability greatly damages that ontology's ability to be reused elsewhere. In practice, this requires manual excision of such statements from all imported ontologies. Interoperability can then be taken care of if needs be, at a global level, with that higher-level scope allowing patches to be applied and developed that will not interfere with other libraries tool-driven patches. To avoid this requirement in future, without discouraging the helpful work that ontology publishers contribute to allowing tools to work with their vocabularies, it would be much better if such patches were distributed as appendices to ontologies rather than incorporated into them.

Equivalence and Hijacking

From the ontologies surveyed, it appears that equivalence within a given ontology is rarely needed. If a class is the same as another class, it seems unlikely to be the case that the ontology designer does not know it. If two classes are indeed the same, it is best to combine the definitions of the classes into a single class, which improves referential transparency and simplifies ontology management. If two names are needed, simply assigning more than one rdfs:label is recommended as a better solution.

However, there is the further use of identification of one class with that of another ontology. Such identification of classes with other ontologies leads to the question of why one would simply not use the class name from the alternative ontology unless one wants to actually hijack the class for

[5] https://smiy.wordpress.com/2010/07/15/the-ordered-list-ontology/
[6] http://owl.cs.manchester.ac.uk/wp-content/uploads/2015/07/list.owl_.txt

extension? And if it is the later, then it seems unfair that the contract be entirely one sided, as any published linked data which comes from the ontology will no longer have the same meaning as that given in the original ontology.

One potential answer to this problem is that ontologies which intend to coordinate, and actually mean to be equivalent, utilise subclassing in either direction. So for instance, instead of saying:

ex:Tome owl:EquivalentClass library:Book

One could say, in the **ex** and **library** ontologies respectively:

ex:Tome rdfs:subClassOf library:Book

library:Book rdfs:subClassOf ex:Tome

In this scenario, collaboration between ontology designers would be required, such that hijacking was less of a concern.

8. Conclusions and Future Work

We have shown that is effective to pursue a reasoner-based approach to detect logical or syntactic errors in linked data schemata based on unified logical models. We have made a first study of the prevalence of errors in the schemata that are most prevalent in the web of data by analysing 62 common vocabulary or ontology specifications. Our validation approach detected 27 errors, 171 instances of ontology namespace violations and 8 vocabularies with dependencies on missing ontologies.

Our analysis began with the practical concern of using Open Annotation (OA) as infrastructure for our own ontology development. After producing a software tool-chain which included ontology management and reasoning, we were able to proceed to testing our ontology's dependence on OA and all of the ontologies which it makes reference to. The results of our survey give valuable information about the state of ontology development, the relative lack of interoperability including the free mixing of ontological frameworks which are logically incompatible, and the fact that tool-chain development is at a very low level, since we believe that many of the problems which we found would have been identified already were it not the case.

We make a number of recommendations regarding how to deal with the realities of ontologies as they currently exist, and how to use them in conjunction with reasoning tool-chains. We detail mechanisms for "patching" existing ontologies in minimal ways such that they are suitable for reasoning, a step which can be taken by practitioners which would like to use these ontologies, but which would ideally be folded into the original ontologies themselves.

We also note the fairly widespread use of higher order features used for meta-modelling, and suggest a way to include such features in a sound fashion free of paradoxes. We hope to explore the consequences of adding stratification to OWL DL and the decidability and complexity consequences thereof in the future.

The utilisation of rdf:List in OWL ontologies really has to be eliminated as it leads to incoherence and the incapacity to reason. In the future, we hope to develop a drop in replacement ontology for RDF collections defined in OWL DL exclusively.

We will be extending our reasoner to include a larger fragment of OWL DL. Our system has already proved useful in finding errors and contains the majority of OWL descriptions which we found in the ontologies explored. A larger fragment should improve the usefulness as it extends the reasoning

facility to a greater class of ontologies. Further, we will be testing our reasoner against ontologies which have extant instance data, and this is likely to reveal more problems than the ones detailed here which are exclusively at the schema level.

Acknowledgement

This research has received funding from the European Union's Horizon 2020 research and innovation programme under grant agreement No 644055, the ALIGNED project (www.aligned-project.eu) and from the ADAPT Centre for Digital Content Technology, funded under the SFI Research Centres Programme (Grant 13/RC/2106) and co-funded by the European Regional Development Fund..

References

[Bizer07] Chris Bizer. Quality-Driven Information Filtering in the Context of Web-Based Information Systems. PhD thesis, Freie Universität Berlin, March 2007.

[Courtot09] Mélanie Courtot, Frank Gibson, Allyson L. Lister, James Malone, Daniel Schober, Ryan R. Brinkman, Alan Ruttenberg (2009), MIREOT: the Minimum Information to Reference an External Ontology Term, Proc. First International Conference on Biomedical Ontology, 26 July 2009

[DC] M. Nilsson, A. Powell, P. Johnston, A. Naeve (2008), Expressing Dublin Core metadata using the Resource Description Framework (RDF), Dublin Core Metadata Initiative, Available at: http://dublincore.org/documents/dc-rdf/

[Demartini12] Gianluca Demartini, Djellel Eddine Difallah, Philippe Cudré-Mauroux, Large-scale linked data integration using probabilistic reasoning and crowdsourcing, The VLDB Journal (2013) 22:665–687, DOI 10.1007/s00778-013-0324-z

[Feeney14] Kevin C. Feeney, Declan O'Sullivan, Wei Tai, and Rob Brennan. Improving curated Web-Data quality with structured harvesting and assessment. Int. J. Semant. Web Inf. Syst., 10(2):35{62, April 2014.

[Furber10] C. Furber and M. Hepp. Using sparql and spin for data quality management on the semantic web. In W. Abramowicz and R. Tolksdorf, editors, BIS, volume 47 of Lecture Notes in Business Information Proccessing , pages 35-46. Springer, 2010.

[Grau08] Bernardo Cuenca Grau, Ian Horrocks, Boris Motik, Bijan Parsia, Peter Patel-Schneider, and Ulrike Sattler. 2008. OWL 2: The next step for OWL. J. Web Semantics. 6, 4 (November 2008), 309-322. DOI=http://dx.doi.org/10.1016/j.websem.2008.05.001

[Gruber93] *Gruber, Thomas Robert (1992). "Toward Principles for the Design of Ontologies Used for Knowledge Sharing" (PDF). International Journal Human-Computer Studies **43**: 907–928.*

[Hepp07] Hepp, M., "Possible Ontologies: How Reality Constrains the Development of Relevant Ontologies," in *Internet Computing, IEEE* , vol.11, no.1, pp.90-96, Jan.-Feb. 2007
doi: 10.1109/MIC.2007.20
URL: http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=4061129&isnumber=4061105

[Heineman 01] G. T. Heineman and W. T. Councill (Eds.). 2001. *Component-Based Software Engineering: Putting the Pieces Together*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA.

[Hogan10] Aidan Hogan, Andreas Harth, Alexandre Passant, Stefan Decker, Axel Polleres, Weaving the pedantic web, in: 3rd International Workshop on Linked Data on the Web, LDOW2010, April 2010.

[Hogan12] A. Hogan, J. Umbrich, A. Harth, R. Cyganiak, A. Polleres, S. Decker, An empirical survey of Linked Data conformance, J. Web Semantics, 14, 14–44, 2012

[Hor03] I. Horrocks, P. F. Patel-Schneider, and F. van Harmelen. From *SHIQ* and RDF to OWL: The Making of a Web Ontology Language. *J. of Web Semantics*, 1(1):7-26, 2003

[Kiryakov09] Atanas Kiryakov, Damyan Ognyanoff, Ruslan Velkov, Zdravko Tashev, Ivan Peikov (2009) LDSR: Materialized Reason-able View to the Web of Linked Data, OWL: Experiences and Directions workshop (OWLED). Chantilly, Virginia, USA, 2009.

[Konto14] Dimitris Kontokostas, Patrick Westphal, Sören Auer, Sebastian Hellmann, Jens Lehmann, Roland Cornelissen, and Amrapali Zaveri. 2014. Test-driven evaluation of linked data quality. In *Proceedings of the 23rd international conference on World wide web* (WWW '14). ACM, New York, NY, USA, 747-758. DOI=http://dx.doi.org/10.1145/2566486.2568002

[Luo90] Z. Luo, An Extended Calculus of Constructions, PhD. Thesis, University of Edinburgh, 1990 Available at: http://www.lfcs.inf.ed.ac.uk/reports/90/ECS-LFCS-90-118/ECS-LFCS-90-118.pdf

[Mendes12] Mendes, P., Muhleisen, H., and Bizer, C. Sieve: Linked data quality assessment and fusion. In LWDM (March 2012).

[MG15] Gavin Mendel-Gleason, Kevin Feeney, Rob Brennan (2015) Ontology Consistency and Instance Checking for Real World Linked Data, Proceedings of the 2nd Workshop on Linked Data Quality co-located with 12th Extended Semantic Web Conference (ESWC 2015), Portorož, Slovenia, June 1, 2015. Available at: http://ceur-ws.org/Vol-1376/LDQ2015_paper_03.pdf

[OA13] Robert Sanderson, Paolo Ciccarese, Herbert Van de Sompel (Eds.), (2013) Open Annotation Data Model, W3C Community Draft, 8 February 2013, Available at: http://www.openannotation.org/spec/core/20130208/

[OWL2 Semantics] B. Motik, P. F. Patel-Schneider, B. C. Grau (Eds.) OWL 2 Web Ontology Language - Direct Semantics (Second Edition), W3C Recommendation 11 December 2012, Available at: http://www.w3.org/TR/2012/REC-owl2-direct-semantics-20121211/

[Patel02] P. F. Patel-Schneider, D. Fensel, Layering the Semantic Web:Problems and Directions, First International Semantic Web Conference (ISWC2002), Sardinia, Italy, June 2002.

[Patel15] Peter F. Patel-Schneider, (2015)Using Description Logics for RDF Constraint Checking and Closed-World Recognition, Twenty-Ninth AAAI Conference on Artificial Intelligence, AAAI-2015, January 25–30, 2015, Austin Texas, USA. Available at: http://arxiv.org/abs/1411.4156v2

[Polleres13] Axel Polleres, Aidan Hogan, Renaud Delbru, and Jurgen Umbrich, (2013) RDFS and OWL Reasoning for Linked Data, Reasoning Web Summer School, Mannheim, 91–149, Springer, 2013. Available at: https://www.deri.ie/sites/default/files/publications/paper.pdf

[Sirin09] E. Sirin and J. Tao. Towards integrity constraints in owl. In Proceedings of the Workshop on OWL:Experiences and Directions, OWLED, 2009.

[TBL01] Tim Berners-Lee, James Hendler and Ora Lassila (2001). "The Semantic Web". Scientific American, May 2001, p. 29-37.

[Bizer09] C. Bizer, T. Heath and T. Berners-Lee (2009) Linked Data - The Story So Far. International Journal on Semantic Web and Information Systems, Vol. 5(3), Pages 1-22. DOI: 10.4018/jswis.2009081901

[Turchin15] Turchin, Peter; Brennan, Rob; Currie, Thomas; Feeney, Kevin; Francois, Pieter; Hoyer, Daniel; et al.(2015). Seshat: The Global History Databank. *Cliodynamics: The Journal of Quantitative History and Cultural Evolution*, 6(1). irows_cliodynamics_27917. Retrieved from: http://escholarship.org/uc/item/9qx38718

[Wielemaker15] Jan Wielemaker, Wouter Beek, Michiel Hildebrand, Jacco van Ossenbruggen, (2015) ClioPatria: A SWI-Prolog infrastructure for the Semantic Web, Semantic Web, vol. Preprint, no. Preprint, pp. 1-13, 2015, DOI: 10.3233/SW-150191

[Zaveri15] Amrapali Zaveri, Anisa Rula, Andrea Maurino, Ricardo Pietrobon, Jens Lehmann and Sören Auer. (2015). Quality assessment for linked data: a survey. Semantic Web. vol. Preprint, no. Preprint, pp. 1-31, 2015 DOI: 10.3233/SW-150175